Description

The present invention relates to a system for processing errors in a bus cycle system.

In such a system, there are normally bus master modules connected to one or more common buses, which bus master modules control the signals on the bus. In a single-chip microcomputer, to which the present invention is particularly but not exclusively directed, a central processing unit (CPU) forms one of the bus master modules, but there are other bus master modules such as e.g. a direct memory access controller (DMAC). For the sake of simplicity, bus master modules other than the CPU will be referred to as "additional" bus master modules.

In a single-chip microcomputer having built-in bus master modules such as a CPU and a DMAC, when a bus error has occurred in a bus cycle started by an additional bus master module, it can be coped with by causing the appropriate additional bus master module to request the CPU to execute an interrupt process corresponding to the bus error.

Such bus errors occur e.g. when there is an attempt to access an address area in which a peripheral circuit or the like is not actually mapped, or when access of a "user" status occurs to a memory area or peripheral device in which a "supervisor" status is set.

In existing systems the bus error ascribable to the additional bus master module has been transacted by the process of interrupt into the CPU as described in, for example, "Signetics, SCC68070, objective specification", page 10, published in December 1985. By way of example, when a bus error has occurred in the bus cycle of a DMAC, the DMAC ends its direct memory access operation and generates a bus error status upon the detection of the bus error. Thereafter, the DMAC releases the mastership of the bus and applies an interupt signal to an interruption controller. The interruption controller gives the CPU an interrupt request for a bus error process within a range allowed according to an interrupt priority level and interrupt mask information. Thus, the CPU verifies the bus error status of the DMAC in accordance with a predetermined interrupt response cycle and runs a service program specific to the bus error process.

In general, there are a plurality of sorts of factors of interrupts, and the various interrupts have their priority levels pre-set in dependence on their predetermined degrees of importance of those factors. For this reason, when an interrupt having a priority level higher than that of an interrupt specific to a bus error has been accepted earlier or where the interrupt for the bus error competes with another interrupt request, the interrupt process for the bus error is rejected and postponed. Even if the bus error is pre-set as a non-maskable interrupt factor, when an exception process or a non-maskable interrupt assigned to another factor has been accepted earlier, an interrupt request of priority level lower than that of the process accepted earlier will not be accepted, with the result that the interrupt process for the bus error does not occur in some cases. Thus, the inventors of the present application have found that when the bus error ascribable to an additional bus master module is processed by the interrupt into the CPU, the reliability of the operation of a system might degrade drastically when the bus cycle of the DMAC or the like is as important as that of the CPU with regard to the operation of the system.

Moreover, even when the interrupt request for the bus error has been accepted by the CPU, the start of the run of a service program for the bus error transaction must be preceded by the decisions of the interrupt factor and the priority level and also by the alteration of the setting of interrupt mask bits for thereafter refusing another interrupt request the priority level of which is lower than that of the interrupt request for the bus error. This poses the problem that a non-negligible time is required before the service program starts to run.

The Motorola M68000 8-/16-/32-Bit Microprocessor User's Manual, seventh edition, PRENTICE HALL, which is a combination of previously published documents, discloses a microprocessor having one Bus Error input. This input causes the current instruction to abort, and exception processing to begin within two clock cycles.

The Abstract of JP-A-59 195 736 discloses an error detecting circuit which could generate a bus error signal.

Hitachi Review, Vol. 35, no. 5, October 1986, page 241-246, Tokyo, Japan discloses an MPU similar to that in the Motorola manual, having a Bus Error input, and a Direct Memory Access Controller to be connected to that MPU. The DMAC disclosed signals bus errors to the MPU by means of an interrupt signal. This combination forms the basis of the precharacterising part of claim 1. Electronic Components and Applications, Vol. 8, no. 2, 1987, pages 95-100, Eindhoven, NL, discloses a similar system to the Hitachi Review article, incorporated on a single chip.

The present invention therefore provides a data processing device comprising a bus (ABUS,DBUS) a central processing unit and a bus master module wherein:

the central processing unit has a first dedicated input for receiving a first bus error detection signal (BERRcpu) and is arranged to execute an exception process program according to the first bus error detection signal (BERRcpu): wherein

the bus master module is arranged to provide an exception process request signal (BETREQ) directly to the central processing unit in response to reception of a second bus error detection signal (BERRdmac) when the bus master module has bus mastership and to release the bus mastership, and

the central processing unit has a second dedicated input for receiving the exception process request signal (BETREQ) from the bus master module and is arranged to execute an exception process program according to the exception process request signal (BETREQ) without judging interrupt mask bits in response to reception of the exception process request signal (BETREQ) by the second dedicated input after the central processing unit acquires the bus mastership owing to the release thereof by the bus master module.

None of the prior art articles discussed above have a dedicated input for detecting bus errors from additional bus master modules.

Thus, according to the present invention, the bus error ascribable to the additional bus master module is set as the specified factor of the exception process for the CPU. In compliance with the exception process request, the CPU starts to run the service program corresponding to the exception process without executing the process for altering and setting mask bits as is executed for an interrupt request. Thus, the exception process request specific to the bus error is prevented from being undesirably refused by e.g. an interrupt request accepted before the bus error. Furthermore, it becomes possible to shorten the period of time which is expended before the start of the run of the service program corresponding to the exception process. Accordingly, even when the additional bus master module requests the bus error exception process during the execution of another exception process or the interrupt process by the CPU, the request for this bus error exception process is not hampered and refused by the interrupt process or the other exception process being already executed, and the execution of the exception process for the bus error of the additional bus master module can be started preferentially.

The period of time which is expended between the request for the bus error exception process from the additional bus master module and the start of the running of the corresponding service program may be shorter than occurs when the bus error is processed with an interrupt process.

Consequently, even when a data transfer operation by the additional bus master module is as important as the bus cycle of the CPU with respect to the operation of the system, such a process for the bus error of the additional bus master module is executed by the exception process in a similar manner to the process for the bus error of the CPU. The result is that the reliability of the system operation can be enhanced.

Furthermore, when processing a bus error of the additional bus master module (which has occurred when the CPU has given a bus mastership to the additional bus master module in the course of the execution of an instruction), the CPU acquires the bus mastership upon such occurrence of the bus error of the bus master module and executes the remaining processing of the instruction, and it thereafter accepts the exception process request, thereby to prevent the loss of a process half executed by the CPU can be prevented from wasting.

In addition, when the request for the exception process specific to the bus error of the additional bus master module is applied through a dedicated signal line to the dedicated terminal of the CPU assigned to this request by, for example, forming the CPU and the additional bus master module on an identical semiconductor substrate, the CPU need not analyze the exception process request thereby making it possible further to shorten the time which is expended until the service program for the exception process for the bus error of the bus master module is carried into run.

An embodiment of the invention will now be described in detail, by way of example, with reference to the accompanying drawings, in which:

Fig. 1 is a block diagram of an embodiment of a single-chip microcomputer according to the present invention;

Fig. 2 is a flow chart showing an example of the operation of transacting the bus error of a direct memory access controller (DMAC) at a specified exception process request; and

Fig. 3 is a flow chart showing an example of the operation of transacting the bus error of the DMAC at an interrupt request.

Fig. 1 shows a single-chip microcomputer which is an embodiment of the present invention. Although the invention is not limited to this, the single-chip microcomputer MCU depicted in the figure is formed on a single semiconductor substrate by known techniques for manufacturing semiconductor integrated circuits.

The single-chip microcomputer MCU shown in Fig. 1 includes a central processing unit (CPU) 1 which is coupled to an address bus ABUS and a data bus DBUS. Also, as an example of additional bus master module (i.e. modules other than the CPU), it includes a direct memory access controller (DMAC) which is coupled to the address bus ABUS and the data bus DBUS. Furthermore, although not shown, bus slave modules such as a program memory for storing a program and an input/output circuit are coupled to the address bus ABUS and the data bus DBUS.

The CPU 1 may include a general register, a program counter, a status register, an arithmetic-logic unit, etc., and it fetches instructions principally from a program memory (not shown) and executes predetermined data processes in succession.

On the basis of bus cycles started by itself, the DMAC 2 transfers a data block between memories and also transfers data between it and a disk controller, a communication interface controller, etc. in con-

formity with a cycle-stealing system or a burst system.

According to this embodiment, a bus arbiter 3 arbitrates a bus mastership between the CPU 1 and the DMAC 2. The bus arbiter 3 arbitrates a request for the bus mastership on the basis of a bus request signal $BREQ_{cpu}$ delivered from the CPU 1 and a bus request signal $BREQ_{dmac}$ delivered from the DMAC 2. It gives the bus mastership to the CPU 1 by asserting a bus acknowledge signal $BACK_{cpu}$, and to the DMAC 2 by asserting a bus acknowledge signal $BACK_{dmac}$. Although not especially restricted, the bus request signals $BREQ_{cpu}$ and $BREQ_{dmac}$ and the bus acknowledge signals $BACK_{cpu}$ and $BACK_{dmac}$ have a high level set as the request level and acknowledge level thereof, respectively.

In Fig. 1, numeral 4 indicates a chip select controller, which is a circuit that forms chip select signals for selecting peripheral circuits allocated to an address space to be managed by the single-chip microcomputer MCU. Although not especially restricted, the chip select controller 4 forms four chip select signals CS1 - CS4 each having a high level as its select level. Each of the chip select signals is directed to one set consisting of an area base register 5, an area range register 6 and a comparator 7. By way of example, the start address of an address area allocated to one predetermined peripheral circuit is set in the area base register 5, while the range of the address area allocated to the predetermined peripheral circuit is set in the area range register 6. The comparator 7 compares an address on the address bus ABUS with addresses within the range designated by the area base register 5 as well as the area range reigster 6, and it brings the corresponding chip select signal to the select level (high level) when the address on the bus is included within the range. Incidentally, in a case where five or more peripheral circuits are mapped in the address space managed by the single-chip microcomputer MCU, a decoder which decodes address signals outside the microcomputer MCU so as to form chip select signals is further required. Besides, in a case where the chip select signals CS1 - CS4 are delivered outside the microcomputer MCU, they can be directly transferred out. Alternatively, the chip select signals are once encoded into 2-bit signals inside the microcomputer MCU so as to externally supply these 2-bit signals, and outside the microcomputer MCU, the supplied signals are decoded into 4-bit signals, which can be utilized.

In Fig. 1, numeral 8 indicates an interruption controller, which is supplied with various interrupt signals NMI1, NMI2, and INT1 - INTn from predetermined peripheral circuits disposed inside and outside the single-chip microcomputer MCU. In this embodiment, the interrupt signals NMI1 and NMI2 are so-called non-maskable interrupt signals and are given the highest priority level in the various interrupt signals. In contrast, the interrupt signals INT1 - INTn are maskable interrupt signals, and they are given priority levels which lower successively in the order mentioned. Information items on the priority levels are held in the CPU 1.

The status register 20 of the CPU 1 retains interrupt enable bits and interrupt mask bits as control information for interrupt requests. The interrupt enable bits are used as mask information in the case where, even when the interrupt requests have been issued, they are not to be given to the CPU 1 for reasons of the generation sources thereof, and they can be set in one-to-one correspondence with the interrupt signals INT1 - INTn. In a case where, during the execution of an interrupt process or the like by the CPU 1, another interrupt request has been made, the interrupt mask bits are used as mask information for determining whether or not the later interrupt request is accepted. These interrupt mask bits can be respectively set in one-to-one correspondence with the various interrupt signals NMI1, NMI2, and INT1 - INTn.

When an interrupt is requested from inside or outside the microcomputer MCU by any of the interrupt signals NMI1, NMI2, and INT1 - INTn, the interruption controller 8 decides whether or not an interrupt enable bit corresponding to the interrupt signal is enabled, through a logical gate 21 illustrated typically. When the interrupt is enabled, the corresponding interrupt request is applied to a priority level arbitration circuit 22. In a case where a plurality of interrupt requests are simultaneously generated, the priority level arbitration circuit 22 arbitrates and controls them according to the predetermined priority levels so as to accept the request of relatively high priority level. The interrupt request having passed through the priority level arbitration circuit 22 is applied to compare and judge means 23 provided in the CPU 1. The compare and judge means 23 judges whether or not the interrupt request applied from the interruption controller 8 is masked by the interrupt mask bit of the status register 20. When not masked, the interrupt request is applied to the execution unit 24 of the CPU 1. Thus, the CPU 1 stores the states of the program counter and the status register 20, and it also alters and sets the interrupt mask bits of the status register 20 so as not to accept new interrupt requests the priority levels of which are lower than that of the accepted interrupt request. Further, the CPU 1 vectors the start address of a service program conforming to the accepted request and runs the program to execute an interrupt process.

In a case where an interrupt is externally requested by the interrupt signal NMI1 or NMI2, the interrupt mask bit is directly checked by the compare and judge means 23 without the decision of an interrupt enable bit and the arbitration based on the priority levels. Even in the case of the non-maskable interrupt request owing to such an interrupt signal NMI1 or NMI2,

a process for altering and setting interrupt mask bits, similar to the foregoing is executed after a process for storing the states, similar to the foregoing. Thus, even when the interrupt request based on the interrupt signal NMI2 is made during an interrupt process based on the other interrupt signal NMI1 by way of example, it is refused to be accepted through the check of the interrupt mask level.

Next, there will be described a construction for a bus error process in the single-chip microcomputer MCU of this embodiment.

Although no special restriction is meant, this embodiment will be exemplified as to the process of a bus error in which a region where no peripheral circuit is actually mapped in the address space managed by the single-chip microcomputer is accessed.

According to this embodiment, in order to detect such bus errors in bus cycles started by themselves, the CPU 1 and the DMAC 2 as the bus master modules are respectively supplied with the output signal of an AND gate 9 which takes the logical product between the bus mastership acknowledge signal $BACK_{cpu}$ and a bus error occurrence signal BERR having its high level set as an error occurrence level, and with the output signal of an AND gate 10 which takes the logical product between the bus mastership acknowledge signal $BACK_{dmac}$ and the bus error occurrence signal BERR, though they are not especially restrictive. Although not especially restricted, the bus error occurrence signal BERR is formed by the output of an AND gate 11 which takes the logical product between bus error occurrence signals BERR1 and BERR2. The bus error occurrence signal BERR1 is the output signal of an AND gate 12 which receives the inverted level signals of the chip select signals CS1 - CS4 as its 4 inputs and takes the logical product among them. On the other hand, the bus error occurrence signal BERR2 is a signal obtained by taking the logical product among the inverted level signals of respective output signals delivered from a decoder, not shown, which decodes address signals outside the single-chip microcomputer in order to form chip select signals.

Accordingly, in a case where the address signals generated in the bus cycle based on the CPU 1 or the DMAC 2 correspond to the address area having no peripheral circuit actually mapped therein, the bus error occurrence signal BERR is brought to the high level. In a case where such a bus error occurs in the bus cycle started by the CPU 1, this CPU 1 samples the high level output of the AND gate 9 at a predetermined timing, thereby to detect the bus error. On the other hand, in a case where the bus error occurs in the bus cycle started by the DMAC 2, this DMAC 2 samples the high level output of the AND gate 10 at a predetermined timing, thereby to detect the bus error.

A bus error detection signal $BERR_{cpu}$ delivered from the AND gate 9 is assigned as a request for an exception process specific to the bus error of the CPU 1. Accordingly, when the specific exception process is requested by the bus error detection signal $BERR_{cpu}$, the CPU 1 stores the states of the program counter and the status register 20, and it thereafter vectors the start address of a service program assigned to the exception process and runs the program to execute a recover process, a retry process or the like for the bus error.

As soon as the DMAC 2 detects the bus error thereof on the basis of the bus error detection signal $BERR_{dmac}$ delivered from the AND gate 10, it stops a direct memory access operation and asserts a bus error process request signal BETREQ directly to the CPU 1. The bus error transaction request signal BETREQ is assigned as a request for an exception process specific to the CPU 1 concerning the bus error of the DMAC 2. When the specified exception process for the bus error of the DMAC 2 is requested, the CPU 1 stores the states of the program counter and the status register 20 at a predetermined timing after acquiring a bus mastership, whereupon it immediately vectors the start address of a service program assigned to the exception process and runs the program to execute a recover process, a retry process or the like for the bus error.

Here, the exception process of the CPU is given a priority level higher than those of interrupt processes in view of the importance of influence which the factor of the occurrence of the exception process exerts on the system operation. In order to reflect this, a process corresponding to the check of an interrupt mask bit is not executed for the exception process, and a process for altering the setting of interrupt mask bits to be executed for an interrupt request is not performed, either, and after the states are stored, the service program corresponding to the exception process is quickly carried into run. Accordingly, when the exception process for the bus error is requested during the execution of the other exception process or the interrupt process by the CPU 1, the process under execution is interrupted, and the exception process for the bus error is preferentially executed. By the way, in a case where the acceptance of interrupts of lower priority levels is to be refused during the bus error exception process, an instruction which alters the setting of the interrupt mask bits so that all or some of the interrupts may be maskable as needed by the system operation can be included in the service program of the pertinent exception process beforehand. However, even when such an instruction is added, it is not affected at all that the bus error exception process request is not hampered and refused by the interrupt process being already executed or the other exception process, and that a period of time expended since the bus error exception process request till the start of the run of the service program is shortened.

Besides, in processing a bus error which has occurred when the CPU 1 has afforded the bus mastership to the DMAC 2 in the course of the execution of an instruction, the CPU 1 acquires the bus mastership after the occurrence of the bus error of the DMAC 2 and executes the rest process of the pertinent instruction and thereafter accepts a request for the exception process of the bus error lest the process executed halfway by the CPU 1 should waste.

Next, the processing operation of the CPU 1 for a bus error having occurred in the bus cycle of the DMAC 2 in this embodiment will be described by chiefly referring to a flow chart of Fig. 2.

When the CPU 1 is performing an operation with bus access (step STP1), the DMAC 2 requests a bus mastership (step STP2). Then, the CPU 1 once abandons the bus mastership for the purpose of an internal process (step STP3), and the DMAC 2 acquires the bus mastership in response to the abandonment. Thus, the DMAC 2 starts its bus cycle (step STP4). When, on this occasion, a bus error in which the DMAC 2 accesses a peripheral circuit or a memory area not mapped in any address space occurs (step STP5), the bus error detection signal $BERR_{dmac}$ which is output from the AND gate 10 is changed to the high level.

Upon detecting the bus error owing to the change of the bus error detection signal $BERR_{dmac}$ to the high level, the DMAC 2 immediately ends the bus cycle to stop a direct memory access operation (step STP6), and it releases the bus mastership and asserts the bus error process request signal BETREQ to the CPU 1 so as to request a corresponding exception process (step STP7).

On this occasion, the CPU 1 has already ended the internal process and requested the bus mastership by way of example (step STP8). When the CPU 1 acquires the bus mastership owing to the release thereof by the DMAC 2, it executes a rest process in one instruction interrupted halfway from being processed, earlier (step STP9) so as to prevent a previous process from wasting. By way of example, the instruction is assumed to be MOVE instruction. Then, when the CPU 1 has performed a read cycle and once released the bus mastership, resulting in a write cycle left undone, it performs the write cycle. After the CPU 1 has executed the last process in the instruction, it samples the exception process request for the bus error of the DMAC 2 (step STP10).

Thus, the main flow of the CPU 1 is interrupted, and the states of the program counter and the status register 20 on that occasion are stored (step STP11). Immediately, the start address of a service program for the exception process specific to the bus error ascribable to the DMAC 2 is vectored (step STP12), and the run of the service program is started (step STP13). After the end of the run of the service program, the operation of the CPU 1 is returned to the main flow via restoring the states (step STP14).

Here, an operation in the case where the bus error ascribable to the DMAC 2 is assumed to be transacted as one of interrupt factors will be described with reference to Fig. 3.

By way of example, it is assumed that an interrupt factor corresponding to an interrupt signal INTi is the bus error ascribable to the DMAC 2. In this case, when an interrupt is requested with the interrupt signal INTi (step STP20), the interruption controller 8 judges whether or not an interrupt enable bit corresponding to the interrupt signal INTi is enabled, through the logical gate 21 (step STP21). When the interrupt is enabled, the interruption controller 8 applies the corresponding interrupt request to the priority level arbitration circuit 22. In a case where another interrupt request is simultaneously issued on this occasion, the priority level arbitration circuit 22 arbitrates and controls the interrupt requests in accordance with predetermined priority levels so as to accept the request of relatively high priority level (step STP22). Accordingly, when the other interrupt the priority level of which is higher than that of the interrupt signal INTi is simultaneously requested, the interrupt request for the process of the bus error ascribable to the DMAC 2 is refused.

Assuming the interrupt signal INTi has passed through the priority level arbitration circuit 22, the interrupt request is applied to the compare and judge means 23 disposed in the CPU 1, and it is judged as to whether or not it is masked by the interrupt mask bit of the status register 20 (step STP23). In a case where the interrupt request is judged to be masked, namely, where the other interrupt or the like at the priority level higher than that of the interrupt signal INTi has already been accepted, the interrupt request for the transaction of the bus error ascribable to the DMAC is refused.

In a case where the judged result of the step STP23 indicates that the interrupt mask bit corresponding to the interrupt signal INTi is disabled, the pertinent interrupt request is applied to the execution unit 24 of the CPU 1, whereby the interrupt request for the bus error process attributed to the DMAC 2 is accepted by the CPU 1 for the first time (step STP24).

Upon the acceptance of the interrupt request, the CPU 1 interrupts the execution of its main flow and stores the states of the program counter and the status register 20 (step STP25). Subsequently, it alters and sets the interrupt mask bits of the status register 20 so as not to accept new interrupt requests having priority levels lower than that of the pertinent interrupt request (step STP26). Then, it vectors the start address of a service program corresponding to the accepted interrupt request (step STP27) and runs the program to start an interrupt process for the bus error of the DMAC 2 (step STP28). After the end of the run of the service program, the operation of the CPU 1 is

returned to the main flow via restoring the states (step STP29).

In addition, assuming that the interrupt factor corresponding to the so-called non-maskable interrupt signal NMI1 is the bus error ascribable to the DMAC 2, the decision of the enable bit at the step STP21 in Fig. 3 and the arbitration based on the priority levels at the step STP22 are not performed, but the check of the interrupt mask bit by the compare and judge means 23 is directly performed (step STP23). Likewise to interrupt requests based on other interrupt signals, even such a non-maskable interrupt request based on the interrupt signal NMI1 undergoes a process for altering and setting interrupt mask bits, similar to the foregoing (step STP26) after the step of storing the states (step STP25). Accordingly, even if the interrupt request based on the interrupt signal NMI1 is made in accordance with the bus error of the DMAC 2 during an interrupt process based on the interrupt signal NMI2 by way of example, it is refused to be accepted by the check of the interrupt mask level.

According to the present invention, the following effects may be attained.

(1) A bus error ascribable to a DMAC 2 is set as the factor of an exeption process specific to a CPU 1, and a request for the bus error exception process is endowed with a priority level higher than that of any interrupt request. For the exception process request, a process corresponding to the check of interrupt mask bits is not performed, and a process for altering the setting of the interrupt mask bits to be executed for the interrupt request is not performed, either, and after states have been stored, a service program corresponding to the exception process is quickly carried into run. More specifically, when the exception process request for the bus error of the DMAC 2 is issued to the CPU 1 by asserting a bus error process request signal BETREQ, the CPU 1 accepts the exception process request directly without executing processes for deciding an enable bit (step STP21), arbitrating interrupt requests on the basis of interrupt priority levels (step STP22) and checking an interrupt mask level (step STP23) as in the case of a process based on an interrupt illustrated in Fig. 3, and after storing states, it carries the service program into run quickly without executing the alteration of the setting of the interrupt mask bits (step STP26). Accordingly, even when the bus error exception process for the DMAC 2 is requested during the execution of another exception process or an interrupt process by the CPU 1, the request for the bus error exception process is not hampered by the interrupt process or the other exception process being already executed and is not refused, so that the execution of the exception process for the bus error of the DMAC 2 can be preferentially started.

(2) Owing to the above effect, a period of time which is expended since the request for the bus error exception process of the DMAC 2 till the start of the run of the service program can be made shorter than in a case where the bus error is processed by an interrupt process.

(3) Owing to the above effects, in a case where a data transfer operation by the DMAC 2 is as important as the bus cycle of the CPU 1 in point of the operation of a system, the reliability of the system operation can be enhanced in consequence of the fact that the bus error of the DMAC 2 is processed by the exception process similarly to the bus error of the CPU 1.

(4) In processing the bus error of the DMAC 2 occurring when the CPU 1 has given a bus mastership to the DMAC 2 in the course of the execution of an instruction, the CPU 1 acquires the bus mastership after the occurrence of the bus error of the DMAC 2 and executes the rest process of the instruction, and it thereafter accepts the exception process request, so that a halfway process which the CPU 1 has executed can be prevented from wasting.

(5) Since the CPU 1 and the DMAC 2 are formed on an identical semiconductor substrate and the exception process request specific to the bus error of the DMAC 2 is applied through a dedicated signal line to the dedicated terminal of the CPU 1 assigned to the pertinent request, the CPU 1 need not analyze the exception process request, thereby making it possible to more shorten the period of time which is expended until the service program for the exception process for the bus error of the DMAC 2 is carried into run.

It has been described in the embodiment that a bus master module other than a CPU is a DMAC, but a bus master module different from the DMAC is similarly permitted to directly supply the CPU with an exception process request for a bus error. Incidentally, the DMAC is not restricted to a single channel, but it may well have a plurality of channels.

Besides, in the embodiment, a bus error in which an address where no peripheral circuit is mapped is output has been explained as an example. However, the bus error is not restricted thereto, but various errors of a bus cycle system, such as an error in which a peripheral circuit or a memory area where a supervisor status is set is accessed in a userstatus, can be processed by exception processes. Further, a system for detecting the bus error is not restricted to that of the embodiment, but it can be properly altered according to the sorts of bus errors.

In addition, a system for vectoring the start address of a service program for the exception process may be any of an indirect vector designating system which employs a vector table, a system in which the

bus master module such as the DMAC affords a vector directly to the CPU, and so on.

In the above, the invention made by the inventors has been chiefly described as to the case of application to a bus error transaction in a single-chip microcomputer having the built-in DMAC as forms the background field of utilization thereof. However, the present invention is not restricted to the application, but it can also be applied to a system in which bus master modules such as a CPU and a DMAC are formed on separate chips, etc. The present invention is applicable to devices each of which has, at least, the condition of transacting a bus error ascribable to a bus master module other than a CPU.

Claims

1. A data processing device comprising a bus (ABUS,DBUS) a central processing unit (1) and a bus master module wherein:

   the central processing unit (1) has a first dedicated input for receiving a first bus error detection signal (BERRcpu) and is arranged to execute an exception process program according to the first bus error detection signal (BERRcpu);

   characterised in that:

   the bus master module (2) is arranged to provide an exception process request signal (BETREQ) directly to the central processing unit (1) in response to reception of a second bus error detection signal (BERRdmac) when the bus master module (2) has bus mastership and to release the bus mastership, and

   the central processing unit (1) has a second dedicated input for receiving the exception process request signal (BETREQ) from the bus master module (2) and is arranged to execute an exception process program according to the exception process request signal (BETREQ) without judging interrupt mask bits in response to reception of the exception process request signal (BETREQ) by the second dedicated input after the central processing unit (1) acquires the bus mastership owing to the release thereof by the bus master module (2).

2. A data processing device according to claim 1, further comprising an interruption controller (8) coupled to the central processing unit (1) and having a plurality of inputs for receiving a plurality of interrupt request signals (INT1-NMI2), the central processing unit (1) having a status register (20) containing the interrupt mask bits for storing mask information indicating a priority level of an interrupt request which the central processing unit (1) is to execute in response to reception of one of the plurality of interrupt request signals, the central processing unit (1) being arranged to set the interrupt mask bits in response to acceptance of the interrupt request, to execute an interrupt process program, and to execute the exception process program according to either bus error detection signal without setting interrupt mask bits.

3. A data processing device according to any one of the preceding claims, wherein the central processing unit (1) is arranged to execute a remaining part of a process of an instruction before the central processing unit (1) executes the exception process program if the central processing unit (1) has abandoned the bus mastership in the course of the execution of the instruction.

4. A data processing device according to any one of the preceding claims, wherein the central processing unit (1) and the bus master module (2) are formed on a single semiconductor chip.

5. A data processing device according to claim 1, further comprising:

   a bus arbiter (3) coupled to the central processing unit (1) and to the bus master module (2) for arbitrating bus mastership between the central processing unit (1) and the bus master module (2), the bus arbiter (3) being arranged to provide a bus acknowledge signal (BACKdmac) to the bus master module (2) such as to give the bus master module (2) the bus mastership;

   wherein the device is arranged so that the second bus error detection signal (BERRdmac) is applied to the bus master module (2) only when the bus arbiter (3) provides the bus acknowledge signal (BACKdmac) to the bus master module (2).

6. A data processing device according to claim 5, wherein the central processing unit (1) the bus master module (2) and the bus arbiter (3) are formed on a single semiconductor chip.

7. A data processing device according to any one of the preceding claims, having means for applying the first bus error detection signal (BERRcpu) to the central processing unit (1) only when the bus arbiter (3) provides a bus acknowledge signal (BACKcpu) to the central processing unit (1).

8. A data processing device according to claim 2, wherein the judging of interrupt mask bits includes setting the interrupt mask bits in response to acceptance of the interrupt request, arranged to execute the exception process program according to either bus error detection signal without setting the interrupt mask bits.

bus master module such as the DMAC affords a vector directly to the CPU, and so on.

In the above, the invention made by the inventors has been chiefly described as to the case of application to a bus error transaction in a single-chip microcomputer having the built-in DMAC as forms the background field of utilization thereof. However, the present invention is not restricted to the application, but it can also be applied to a system in which bus master modules such as a CPU and a DMAC are formed on separate chips, etc. The present invention is applicable to devices each of which has, at least, the condition of transacting a bus error ascribable to a bus master module other than a CPU.

Claims

1. A data processing device comprising a bus (ABUS,DBUS) a central processing unit (1) and a bus master module wherein:

the central processing unit (1) has a first dedicated input for receiving a first bus error detection signal (BERRcpu) and is arranged to execute an exception process program according to the first bus error detection signal (BERRcpu);

characterised in that:

the bus master module (2) is arranged to provide an exception process request signal (BETREQ) directly to the central processing unit (1) in response to reception of a second bus error detection signal (BERRdmac) when the bus master module (2) has bus mastership and to release the bus mastership, and

the central processing unit (1) has a second dedicated input for receiving the exception process request signal (BETREQ) from the bus master module (2) and is arranged to execute an exception process program according to the exception process request signal (BETREQ) without judging interrupt mask bits in response to reception of the exception process request signal (BETREQ) by the second dedicated input after the central processing unit (1) acquires the bus mastership owing to the release thereof by the bus master module (2).

2. A data processing device according to claim 1, further comprising an interruption controller (8) coupled to the central processing unit (1) and having a plurality of inputs for receiving a plurality of interrupt request signals (INT1-NMI2), the central processing unit (1) having a status register (20) containing the interrupt mask bits for storing mask information indicating a priority level of an interrupt request which the central processing unit (1) is to execute in response to reception of one of the plurality of interrupt request signals, the central processing unit (1) being arranged to set the interrupt mask bits in response to acceptance of the interrupt request, to execute an interrupt process program, and to execute the exception process program according to either bus error detection signal without setting interrupt mask bits.

3. A data processing device according to any one of the preceding claims, wherein the central processing unit (1) is arranged to execute a remaining part of a process of an instruction before the central processing unit (1) executes the exception process program if the central processing unit (1) has abandoned the bus mastership in the course of the execution of the instruction.

4. A data processing device according to any one of the preceding claims, wherein the central processing unit (1) and the bus master module (2) are formed on a single semiconductor chip.

5. A data processing device according to claim 1, further comprising:

a bus arbiter (3) coupled to the central processing unit (1) and to the bus master module (2) for arbitrating bus mastership between the central processing unit (1) and the bus master module (2), the bus arbiter (3) being arranged to provide a bus acknowledge signal (BACKdmac) to the bus master module (2) such as to give the bus master module (2) the bus mastership;

wherein the device is arranged so that the second bus error detection signal (BERRdmac) is applied to the bus master module (2) only when the bus arbiter (3) provides the bus acknowledge signal (BACKdmac) to the bus master module (2).

6. A data processing device according to claim 5, wherein the central processing unit (1) the bus master module (2) and the bus arbiter (3) are formed on a single semiconductor chip.

7. A data processing device according to any one of the preceding claims, having means for applying the first bus error detection signal (BERRcpu) to the central processing unit (1) only when the bus arbiter (3) provides a bus acknowledge signal (BACKcpu) to the central processing unit (1).

8. A data processing device according to claim 2, wherein the judging of interrupt mask bits includes setting the interrupt mask bits in response to acceptance of the interrupt request, arranged to execute the exception process program according to either bus error detection signal without setting the interrupt mask bits.

Patentansprüche

1. Datenverarbeitungsvorrichtung mit einem Bus (ABUS, DBUS), einer zentralen Verarbeitungseinheit (1) und einem Busmastermodul, wobei:
   - die zentrale Verarbeitungseinheit (1) einen ersten reservierten Eingang zum Empfangen eines ersten Busfehler-Erkennungssignals (BERRcpu) aufweist und sie so ausgebildet ist, daß sie abhängig vom ersten Busfehler-Erkennungssignal (BERRcpu) ein Ausnahmeprozeßprogramm ausführt;

   dadurch gekennzeichnet, daß
   - das Busmastermodul (2) so ausgebildet ist, daß es auf Empfang eines zweiten Busfehler-Erkennungssignals (BERRdmac) hin ein Ausnahmeprozeß-Anforderungssignal (BETREQ) direkt an die zentrale Verarbeitungseinheit (1) liefert, wenn es die Busherrschaft hat, und es diese Busherrschaft freigibt; und
   - die zentrale Verarbeitungseinheit (1) einen zweiten reservierten Eingang zum Empfangen des Ausnahmeprozeß-Anforderungssignals (BETREQ) vom Busmastermodul (2) aufweist und sie so ausgebildet ist, daß sie ein Ausnahmeprozeßprogramm abhängig vom Ausnahmeprozeß-Anforderungssignal (BETREQ) ohne Beurteilung von Interruptmaskenbits auf den Empfang des Ausnahmeprozeß-Anforderungssignals (BETREQ) über den zweiten reservierten Eingang ausführt, nachdem sie die Busherrschaft dadurch erlangt hat, daß diese vom Busmastermodul (2) freigegeben wurde.

2. Datenverarbeitungsvorrichtung nach Anspruch 1, ferner mit einem Interruptcontroller (8), der mit der zentralen Verarbeitungseinheit (1) verbunden ist und mehrere Eingänge zum Empfangen mehrerer Interruptanforderungssignale (INT1 - NMI2) aufweist, wobei die zentrale Verarbeitungseinheit (1) über ein Statusregister (20) verfügt, das Interruptmaskenbits zum Abspeichern von Maskeninformation enthält, die den Prioritätsrang einer Interruptanforderung anzeigt, die die zentrale Verarbeitungseinheit (1) auf den Empfang eines der mehreren Interruptanforderungssignale hin ausführen soll, wobei die zentrale Verarbeitungseinheit (1) so ausgebildet ist, daß sie die Interruptmaskenbits auf den Empfang einer Interruptanforderung hin setzt, um ein Interruptprozeßprogramm auszuführen, und daß sie das Ausnahmeprozeßprogramm auf jedes Busfehler-Erkennungssignal hin ausführt, ohne Interruptmaskenbits zu setzen.

3. Datenverarbeitungsvorrichtung nach einem der vorstehenden Ansprüche, bei der die zentrale Verarbeitungseinheit (1) so ausgebildet ist, daß sie den restlichen Teil eines Prozesses für eine Anweisung ausführt, bevor sie das Ausnahmeprozeßprogramm ausführt, wenn sie die Busherrschaft im Verlauf der Ausführung der Anweisung aufgegeben hat.

4. Datenverarbeitungsvorrichtung nach einem der vorstehenden Ansprüche, bei der die zentrale Verarbeitungseinheit (1) und das Busmastermodul (2) auf einem einzigen Halbleiterchip ausgebildet sind.

5. Datenverarbeitungsvorrichtung nach Anspruch 1, ferner mit:
   - einem Busprioritätszuteiler (3), der mit der zentralen Verarbeitungseinheit (1) und dem Busmastermodul (2) verbunden ist, um die Priorität für die Busherrschaft zwischen der zentralen Verarbeitungseinheit (1) und dem Busmastermodul (2) zuzuteilen, und der so ausgebildet ist, daß er ein Busbestätigungssignal (BACKdmac) an das Busmastermodul (2) liefert, um dem Busmastermodul (2) die Busherrschaft zu übergeben;
   - wobei die Vorrichtung so ausgebildet ist, daß das zweite Busfehler-Erkennungssignal (BERRdmac) nur dann an das Busmastermodul (2) gegeben wird, wenn der Busprioritätszuteiler (3) das Busbestätigungssignal (BACKdmac) an das Busmastermodul (2) liefert.

6. Datenverarbeitungsvorrichtung nach Anspruch 5, bei der die zentrale Verarbeitungseinheit (1), das Busmastermodul (2) und der Busprioritätszuteiler (3) auf einem einzigen Halbleiterchip ausgebildet sind.

7. Datenverarbeitungsvorrichtung nach einem der vorstehenden Ansprüche, mit einer Einrichtung zum Anlegen des ersten Busfehler-Erkennungssignals (BERRcpu) an die zentrale Verarbeitungseinheit (1) nur dann, wenn der Busprioritätszuteiler (3) ein Busbestätigungssignal (BACKcpu) an die zentrale Verarbeitungseinheit (1) liefert.

8. Datenverarbeitungsvorrichtung nach Anspruch 2, bei der zur Beurteilung der Interruptmaskenbits das Setzen der Interruptmaskenbits auf den Empfang einer Interruptanforderung hin gehört, mit einer solchen Ausbildung, daß das Ausnahmeprozeßprogramm auf jedes Busfehler-Erkennungssignal hin ausgeführt wird, ohne die Interruptmaskenbits zu setzen.

Revendications

1. Dispositif de traitement de données comprenant un bus (ABUS,DBUS), une unité centrale de traitement (1) et un module maître de bus, dans lequel :

   l'unité centrale de traitement (1) possède une première entrée spécialisée servant à recevoir un premier signal de détection d'erreur de bus (BERRcpu) et est agencée de manière à exécuter un programme de processus d'exception conformément au premier signal de détection d'erreur de bus (BERRcpu); caractérisé en ce que

   le module maître de bus (2) est agencé de manière à envoyer un signal de demande d'un processus d'exception (BETREQ) directement à l'unité centrale de traitement (1) en réponse à la réception d'un second signal de détection d'erreur de bus (BERRdmac) lorsque le module maître de bus (2) possède la maîtrise du bus, et pour abandonner la maîtrise du bus, et

   l'unité centrale de traitement (1) possède une seconde entrée spécialisée pour la réception du signal de demande de processus d'exception (BETREQ) en provenance du module maître de bus (2) et est agencée de manière à exécuter un programme de processus d'exception conformément au signal de demande de processus d'exception (BETREQ) sans évaluation des bits du masque d'interruption en réponse à la réception du signal de demande de processus d'exception (BETREQ) par la seconde entrée spécialisée, une fois que l'unité centrale de traitement (1) acquiert la maîtrise du bus, en raison de la libération du bus par le module maître de bus (2).

2. Dispositif de traitement de données selon la revendication 1, comprenant en outre un dispositif de commande d'interruption (8) couplé à l'unité centrale de traitement (1) et possédant une pluralité d'entrées pour recevoir une pluralité de signaux de demande d'interruption (INT1-NMI2), l'unité centrale de traitement (1) possédant un registre d'état (20) contenant les bits de masque d'interruption pour mémoriser une information de masque indiquant un niveau de priorité d'une demande d'interruption, que l'unité centrale de traitement (1) doit exécuter en réponse à la réception d'un signal faisant partie d'une pluralité de signaux de demande d'interruption, l'unité centrale de traitement (1) étant agencée de manière à positionner les bits de masque d'interruption en réponse à l'acceptation de la demande d'interruption, pour l'exécution d'un programme de processus d'interruption, et exécuter le programme de processus d'exception conformément à, l'un ou l'autre des signaux de détection d'erreur de bus sans positionnement de bits de masque d'interruption.

3. Dispositif de traitement de données selon l'une quelconque des revendications précédentes, dans lequel l'unité centrale de traitement (1) est agencée de manière à exécuter une partie restante d'un processus d'une instruction avant que l'unité centrale de traitement (1) exécute le programme de processus d'exception, si l'unité centrale de traitement (1) a abandonné la maîtrise du bus au cours de l'exécution de l'instruction.

4. Dispositif de traitement de données selon l'une quelconque des revendications précédentes, dans lequel l'unité centrale de traitement (1) et le module maître de bus (2) sont formés sur une seule microplaquette à semiconducteurs.

5. Dispositif de traitement de données selon la revendication 1, comprenant en outre :

   un dispositif d'arbitrage de bus (3) couplé à l'unité centrale de traitement (1) et au module maître de bus (2) pour arbitrer la maîtrise du bus entre l'unité centrale de traitement (1) et le module maître de bus (2), le dispositif d'arbitrage de bus (3) étant agencé de manière à délivrer un signal d'accusé de réception de bus (BACKdmac) au module maître de bus (2) de manière à conférer la maîtrise du bus au module maître de bus (2);

   le dispositif étant agencé de telle sorte que le second signal de détection d'erreur de bus (BERRdmac) est appliqué au module maître de bus (2) uniquement lorsque le dispositif d'arbitrage de bus (3) délivre le signal d'accusé de réception de bus (BACKdmac) au module maître de bus (2).

6. Dispositif de traitement de données selon la revendication 5, dans lequel l'unité centrale de traitement (1), le module maître de bus (2) et le dispositif d'arbitrage de bus (3) sont formés sur une seule microplaquette à semiconducteurs.

7. Dispositif de traitement de données selon l'une quelconque des revendications précédentes, comportant des moyens pour appliquer le premier signal de détection d'erreur de bus (BERRcpu) à l'unité centrale de traitement (1) uniquement lorsque le dispositif d'arbitrage de bus (3) envoie un signal d'accusé de réception de bus (BACKcpu) à l'unité centrale de traitement (1).

8. Dispositif de traitement de données selon la revendication 2, dans lequel l'évaluation de bits de masque et d'interruption comprend le positionnement des bits de masque d'interruption en répon-

* C L M S *

se à l'acceptation de la demande d'interruption, agencé de manière à exécuter le programme de processus d'exception conformément à l'un ou l'autre des signaux de détection d'erreur de bus sans positionnement des bits de masque d'interruption.